FFT Butterfly using carrysave

- **FFT butterfly equations in complex form**

  Z1 = A + W*B

  Z2 = A - W*B

- **Real and Imaginary equations**

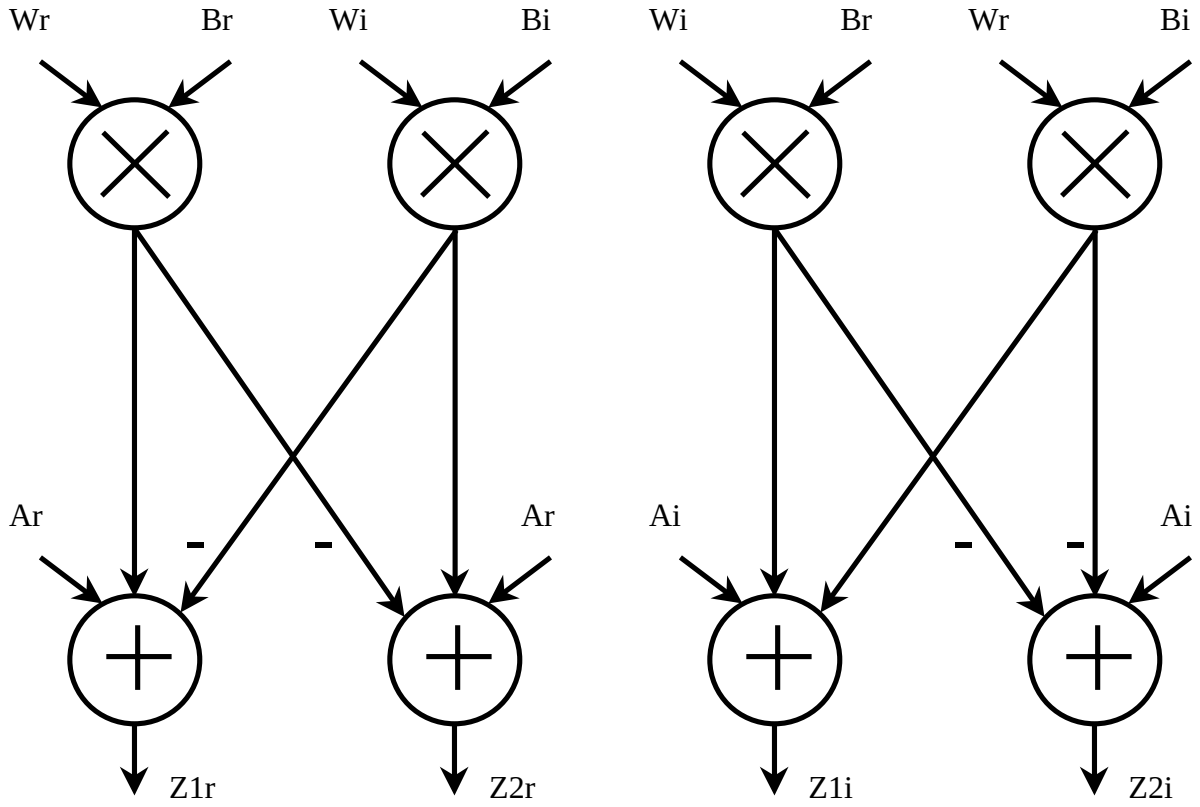
  Z1r = Ar + Wr*Br - Wi*Bi

  Z1i = Ai + Wi*Br + Wr*Bi

  Z2r = Ar - Wr*Br + Wi*Bi

  Z2i = Ai - Wi*Br - Wr*Bi

FFT Butterfly using carrysave

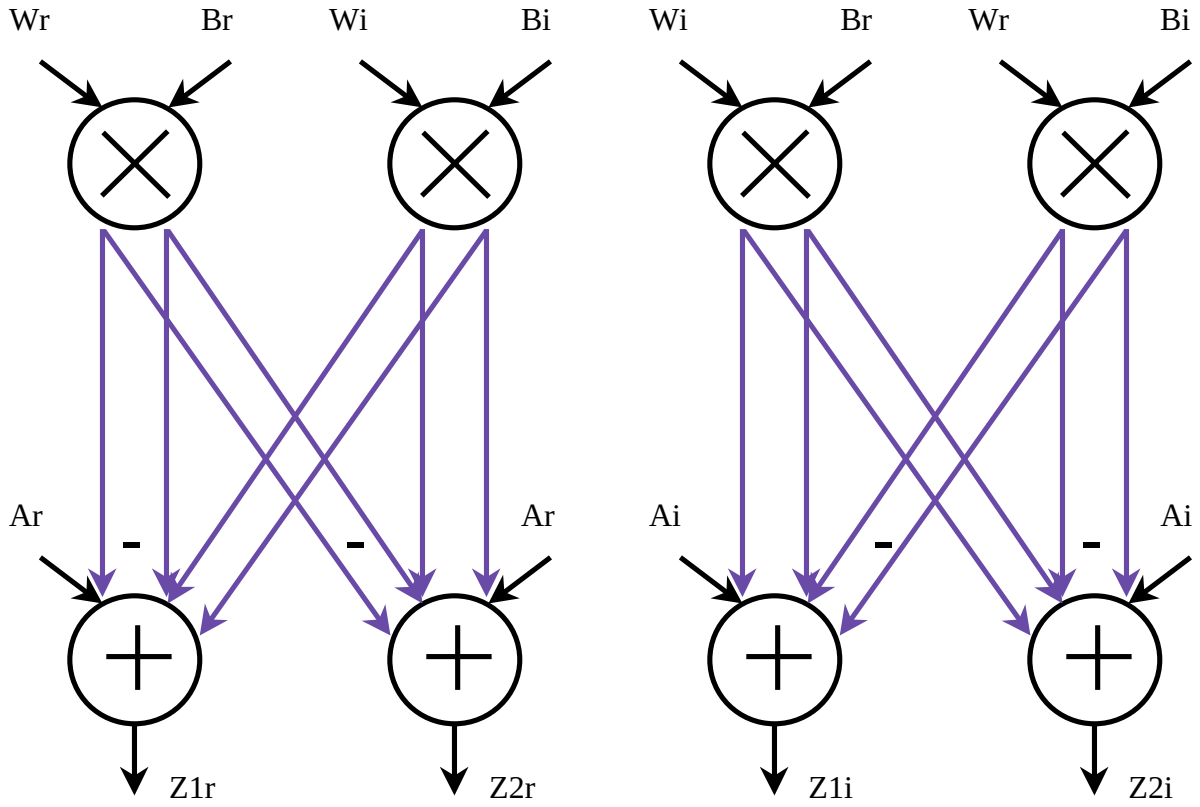


operator merging
only one carry propagate adder

FFT Butterfly using carrysave

outputs in redundant binary form

operator merging with redundant binary signals



FFT Butterfly using carrysave

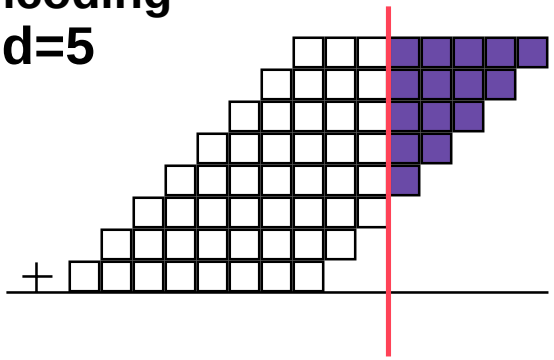
```
// 4 product terms left in carrysave format -- no carry propagate add
directive (carrysave="on");
wire [2*w] P1 = Wr*Br;
wire [2*w] P2 = Wi*Bi;
wire [2*w] P3 = Wi*Br;
wire [2*w] P4 = Wr*Bi;

// 4 result terms in binary format -- 4 carry propagate adders
directive (carrysave="off");
wire [2*w] Z1r = (Ar<<w) + P1 - P2;
wire [2*w] Z2r = (Ar<<w) - P1 + P2;
wire [2*w] Z1i = (Ai<<w) + P3 + P4;
wire [2*w] Z2i = (Ai<<w) - P3 - P4;
```

Internal Rounding

- Internal rounding enables a tradeoff between computational accuracy and circuit area
- Simple example -- 8x8 multiplier with unsigned inputs and non-booth encoding

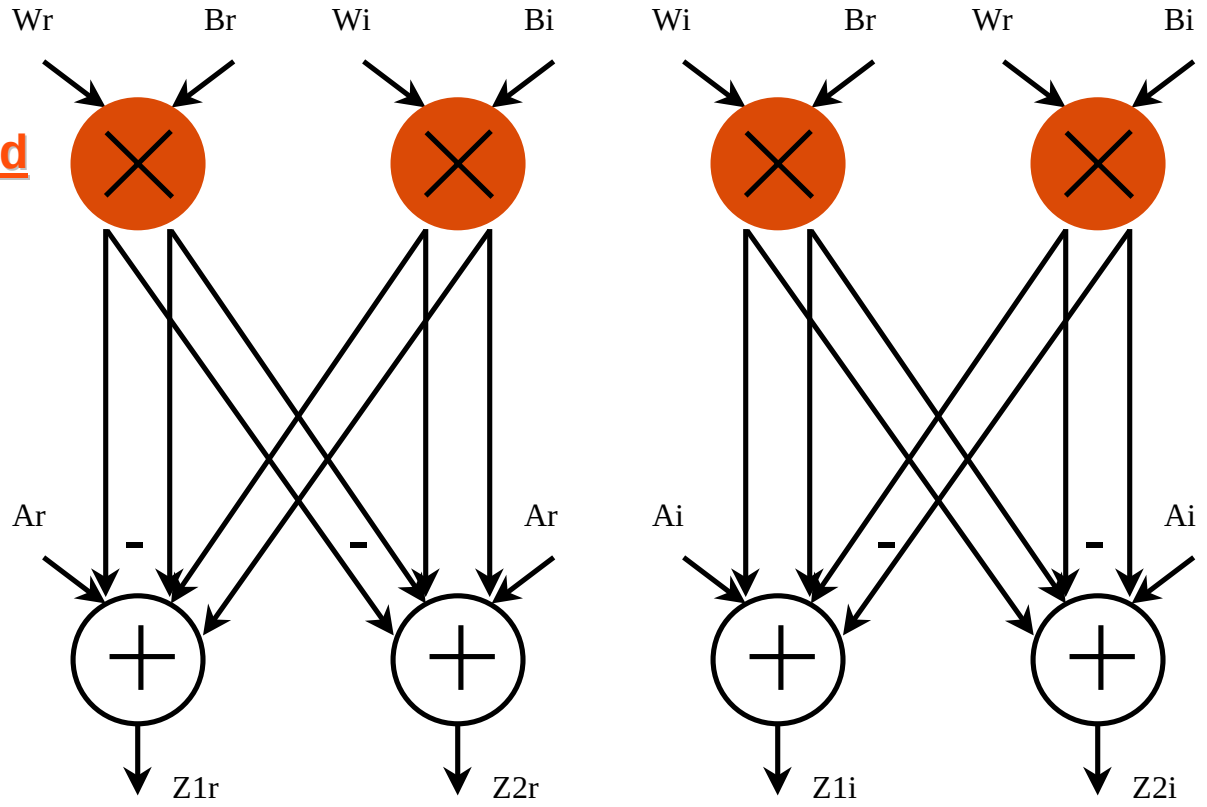
  intround=5



- Internal rounding works with any addition-based expression, e.g., A+B+C-D+ X*Y > L*M - 314*H

FFT Butterfly using carrysave

internally rounded multipliers



FFT Butterfly using carrysave and internal rounding

```
// 4 product terms left in carrysave format -- no carry propagate add
// programmable internal rounding position
directive (carrysave="on",intround=ir);
wire [2*w] P1 = Wr*Br;
wire [2*w] P2 = Wi*Bi;
wire [2*w] P3 = Wi*Br;
wire [2*w] P4 = Wr*Bi;

// 4 result terms in binary format -- 4 carry propagate adders
directive (carrysave="off",intround=0);
wire [2*w] Z1r = (Ar<<w) + P1 - P2;
wire [2*w] Z2r = (Ar<<w) - P1 + P2;
wire [2*w] Z1i = (Ai<<w) + P3 + P4;
wire [2*w] Z2i = (Ai<<w) - P3 - P4;
```